

Analysis of d-Hop Dominating Set Problem for Directed Graph with Indegree Bounded by One

Joydeep Banerjee, Arun Das and Arunabha Sen
 Computer Science and Engineering Program,
 School of Computing, Informatics and Decision System Engineering
 Arizona State University
 Tempe, Arizona 85287
 {Joydeep.Banerjee, adas, asen}@asu.edu

April 29, 2014

Abstract

d-hop dominating set problem was introduced for cluster formation in wireless ad-hoc networks and is proved to be NP-complete. Dominating set problem for directed graph with indegree of at most 1 can be solved polynomially. In this article we perform an analysis of d-hop dominating set problem for directed graph with indegree 1. We show that the problem can be solved polynomially by exploiting certain properties of the graph under consideration.

1 Introduction

d-hop minimum dominating set problem was introduced in [1] for cluster formation in wireless ad-hoc networks. The problem is proved to be NP complete. The decision version of minimum d-hop dominating set problem is given as follows:

Instance: An undirected graph $G = (V, E)$, two positive integers d and K .

Question: Is there a subset $|V'| \subseteq V$ with $|V'| \leq K$ such that for every vertex $v \notin V - V'$ is at most d hops away from at least one vertex in V' .

In a directed graph $G = (V, E)$ a node u dominates a node v if the edge $(u, v) \in E$. For a directed graph dominating set is proved to NP complete[2]. But for directed graph with indegree of at most 1, the problem can be solved polynomially [2].

In this article we analyze the problem *d-hop minimum dominating set problem for directed graph with indegree bounded by 1*. We show the existence of a poly-

nomial time algorithm of the problem by exploiting certain properties of the graph under consideration.

2 Analysis of the Problem

Certain properties and definitions of a directed graph with indegree bounded by one (denoted as the graph $G_D = (V_D, E_D)$) is introduced before analysis of the problem.

Property 2.1 *Each weakly connected subgraph of the directed graph G_D consists of at most one cycle with no incoming edge to a node in the cycle from a node not in the cycle.*

Thus a weakly connected component is either a Directed Acyclic Graph (DAG) or has a cycle with all nodes not in the cycle having a directed path from all nodes in the cycle to it.

Definition 2.1 *A leaf node of a weakly connected component is defined as the node with an incoming edge and no outgoing edge.*

Definition 2.2 *The farthest leaf node of a weakly connected component is defined as the node which is at furthest hops from the root node (if the graph is a DAG) or from a node in the cycle.*

Definition 2.3 *An isolated strongly connected component is defined as the component which is not a subgraph of a weakly connected component of the graph G_D*

Property 2.2 *Each isolated strongly connected subgraph of the directed graph G_D is composed of exactly one cycle.*

Exploiting the properties of the graph G_D an algorithm (Algorithm 1) is framed. The proof of optimality and time complexity analysis of the algorithm are done in Theorem 1 and Theorem 2 respectively.

Theorem 2.1 *Algorithm 1 gives the optimum solution of d -hop dominating set problem for the graph G_D .*

Proof For an isolated strongly connected component $G_S = (V_S, E_S)$ at least $\lceil \frac{|V_S|}{d} \rceil$ nodes has to be included in the solution. Algorithm 1 selects $\lceil \frac{|V_S|}{d} \rceil$ nodes with at least $\lceil \frac{|V_S|}{d} \rceil - 1$ nodes separated by exactly d hops. Thus it includes the optimum nodes in the solution. For a weakly connected component a leaf node has to be dominated by a node within d hops away from it. Selecting the farthest leaf node would ensure that after updating the graph G_W with node removals would not result in segregation of the updated graph with more than one weakly connected component. Moreover selecting a leaf node which

Algorithm 1 Algorithm for finding d-hop dominating set of graph G_D

```
Set  $D = \emptyset$ ;  
Compute all weakly connected component and isolated strongly connected  
component of graph  $G_D$ ;  
for (Each weakly connected component  $G_W = (V_W, E_W)$  of graph  $G_D$ ) do  
  while (Graph  $G_W$  is not empty) do  
    if (Graph  $G_W$  is a isolated strongly connected component) then  
      For all  $n$  nodes in graph  $G_W$ , include  $\lceil \frac{n}{d} \rceil$  nodes in set  $D$  with at  
      least  $\lceil \frac{n}{d} \rceil - 1$  nodes separated by exactly  $d$  hops;  
      break;  
    end if  
    Pick the farthest leaf node  $v$  from graph  $G_W$ ;  
    Include node  $u$  in set  $D$  such that the number of hops from  $u$  to  $v$  is  
    maximum but is less than  $d$ ;  
    Update graph  $G_W$  by removing all nodes (and their edges) which are  
    within  $d$  hops from  $u$ , including node  $u$ ;  
  end while  
end for  
for (Each isolated strongly connected component  $G_S = (V_S, E_S)$  of graph  
 $G_D$ ) do  
  Include  $\lceil \frac{|V_S|}{d} \rceil$  nodes in set  $D$  with at least  $\lceil \frac{|V_S|}{d} \rceil - 1$  nodes separated by  
  exactly  $d$  hops;  
end for
```

is not farthest might result in excluding the node which d-hop dominates it along with the farthest leaf node. Thus another node needs to be included in the solution to d-hop dominate the farthest leaf node and so the solution would not be optimum. Hence a node that is maximum hops away from the farthest leaf node (with number of hops less than equal to d) for the weakly connected graph G_W (with or without updating with node removals) has to be included in the optimum solution of the problem. Additionally consider computing the d-hop dominating set for a weakly connected component G_W with one cycle. This may result in an isolated strongly connected component of original graph G_W (i.e. the cycle) after subsequent update of graph G_W with node removals (as in Algorithm 1). The d-hop dominating set of the resultant graph then can be solved similarly to that of strongly connected component, which is optimum. Hence Algorithm 1 computes the optimum d-hop dominating set problem for each weakly connected component and strongly connected component of graph G_D polynomially.

Theorem 2.2 *Algorithm 1 solves d-hop dominating set problem for the graph G_D polynomially with time complexity of order \mathcal{O}*

Proof All strongly connected components can be found using Tarjan's strongly connected components algorithm [3] in $\mathcal{O}(|V_D| + |E_D|)$ time. The isolated

strongly connected component from the computed strongly connected components can be found in another $\mathcal{O}(|V_D|)$ time. Hence computing the isolated strongly connected component takes $\mathcal{O}(|V_D|(|V_D| + |E_D|))$ time. Deleting all nodes in the strongly connected component would result in segregation of the graph G_D into weakly connected components. All weakly connected components can be found by computing existence of pairwise path between two nodes. It can be done in $\mathcal{O}(|V_D| + |E_D|)$ time for each pair and each weakly connected component can be computed in $\mathcal{O}(|V_D|(|V_D| + |E_D|))$ time. Both the first for loop and the while loop inside it iterates for at most $\mathcal{O}(|V_D|)$ times. The operation inside the if branch can be computed $\mathcal{O}(|V_D|)$ time. The farthest leaf node can be found in $\mathcal{O}(|V_D|^2)$ time. The d-hop dominating node of the farthest leaf node can be found in $\mathcal{O}(|V_D|)$ time. Hence the d-hop dominating set computation for all weakly connected component takes $\mathcal{O}(|V_D|^4)$ time. For computing the d-hop dominating set for all strongly connected component, the for loop takes $\mathcal{O}(|V_D|)$ time. The computation inside the for loop is same as the computation inside the if branch of the weakly connected component d-hop dominating set computation and takes $\mathcal{O}(|V_D|)$ time. So the total time complexity for finding d-hop dominating set for strongly connected component is $\mathcal{O}(|V_D|^2)$. Hence Algorithm 1 takes in total an $\mathcal{O}(|V_D|^4)$ time to compute the optimum solution of the problem.

3 Conclusion

In this article analysis of d-hop dominating set for directed graph with indegree bounded by one is performed. It is found that exploiting certain properties of the graph under consideration an algorithm can solve the problem in polynomial time, with run time complexity bounded by four times the number of vertices in the graph.

References

- [1] Amis, Alan D., et al. "Max-min d-cluster formation in wireless ad hoc networks." INFOCOM 2000. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE. Vol. 1. IEEE, 2000.
- [2] Albers, Susanne, and Tomasz Radzik, eds. Algorithms–ESA 2004: 12th Annual European Symposium, Bergen, Norway, September 14-17, 2004, Proceedings. Vol. 12. Springer, 2004.
- [3] Tarjan, R. E. "Depth-first search and linear graph algorithms", SIAM Journal on Computing 1 (2): 146160, 1972.